Dynamic Scope-Based Dijkstra's Algorithm*

Petr Hliněný and Ondrej Moriš

Faculty of Informatics, Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
hlineny@fi.muni.cz, xmoris@fi.muni.cz

Abstract. We briefly report on the current state of a new dynamic algorithm for the route planning problem based on a concept of scope (the static variant presented at ESA'11, [16]). We first motivate dynamization of the concept of scope admissibility, and then we briefly describe a modification of the scope-aware query algorithm of [16] to dynamic road networks. Finally, we outline our future work on this concept.

1 Introduction

The single pair shortest path problem in real-world road networks, also known as route planning, has many important everyday applications. There are two most significant variants of the problem – static and dynamic route planning.

In the *static* variant, a road network is fixed during the computation of optimal routes (*query*). Static route planning received a lot of attention during the last decades. On the other hand, in the *dynamic* variant a road network is subject to change in time – some new roads are built, some other are closed, traffic jams or car accidents happen, some routes must be avoided due to turn angle limits, and so on. Clearly, the latter is a more realistic scenario. Actually, even time-dependent cost functions can be modeled in dynamic route planning to some extent.

Classical algorithms such as Dijkstra's [1] or A* [3] and their dynamic adaptions [2] are not well suitable neither for static nor for dynamic variants because of huge road networks size. Thus, a feasible solution lies in computing suitable auxiliary data of the road network (*preprocessing*) in order to improve both time and space complexity of subsequent queries. This technique led to several very interesting static approaches in the last decade, see extensive surveys, for instance, in [13,14]. Some of these algorithms such as highway-hierarchies [10], ALT [9] or, for example, geometric containers [8] were proved to work fine also in the dynamic scenario [7,6,11,12].

Recently, in order to fill a gap between a variety of exact route planning approaches, we have published [16] a different novel approach aimed at "reasonable" routes. It is based on a concept of scope, whose core idea can be informally outlined as follows: The edges of a road network are associated with a *scope* map

* Research supported by the Czech Science Foundation, grants P202/11/0196 and Eurocores GIG/11/E023.

such that an edge e assigned scope s_e is admissible on a route R if, before or after reaching e, such R travels distance less than a value associated with s_e on edges with scope higher than s_e. The desired effect is that low-level roads are fine near the start or target positions, while only roads of the highest scope are admissible in the long middle sections of distant routing queries. Overall, this nicely corresponds with human thinking of intuitive routes, and allows for a very space-efficient preprocessing, too.

New Contribution. In the dynamic scenario, however, a static scope map may badly fail. Imagine, for instance, a closure of a motorway tunnel which can be bypassed only on low-level mountain roads. Then a detour would not be scope admissible in the aforementioned sense, and so a dynamic adjustement of this definition is needed. We present such an adjusted definition here, along with a modification of Dijkstra's algorithm for scope admissible routes in this dynamic scenario. Our algorithm is exact, and its time complexity grows only slightly over ordinary Dijkstra if few negative changes are introduced in the network.

2 Preliminaries

A *directed graph* G is a pair of a finite set $V(G)$ of vertices and a finite multi-set $E(G) \subseteq V(G) \times V(G)$ of edges. A *walk* $P \subseteq G$ is an alternating sequence $(u_0, e_1, u_1, \ldots, e_k, u_k)$ of vertices and edges of G such that $e_i = (u_{i-1}, u_i)$ for $i = 1, \ldots, k$. The *weight* of a walk $P \subseteq G$ w.r.t. a weighting $w : E(G) \mapsto \mathbb{R}$ of G is defined as $|P|_w = w(e_1)+w(e_2)+\cdots+w(e_k)$ where $P = (u_0, e_1, \ldots, e_k, u_k)$. An *optimal walk* between two vertices achieves the minimum weight over all walks.

A road network is referred to as a pair (G, w) where G is a directed graph (such that the junctions are represented by $V(G)$ and the roads by $E(G)$), and w (cost function) is given as a *non-negative* edge weighting $w : E(G) \mapsto \mathbb{R}_0^+$. In the dynamic scenario, w is simply replaced with w^* (differing from w only on few edges, say). If e is removed then let $w^*(e) = \infty$.

Driven by real-world motivation, we focus on negative (increased weight, even to ∞) changes in w. We thus now for simplicity omit the possibility of adding new edges to G, though we understand it may be useful when, e.g., a designated detour locally changes the road network.

3 Scope and Scope Admissibility

A simplified version of the scope concept is briefly introduced here. We strongly recommend reading [16] for more detailed treatment and, due to lack of space, omit most details here.

Definition 3.1 ([16]). *Let (G, w) be a road network.* A scope mapping *is defined as $\mathcal{S} : E(G) \mapsto \mathbb{N}_0 \cup \{\infty\}$ such that $0, \infty \in Im(\mathcal{S})$. Elements of the image $Im(\mathcal{S})$ are called* scope levels. *Each scope level $i \in Im(\mathcal{S})$ is assigned a constant value of* scope $\nu_i^{\mathcal{S}} \in \mathbb{R}_0 \cup \{\infty\}$ *such that $0 = \nu_0^{\mathcal{S}} < \nu_1^{\mathcal{S}} < \cdots < \nu_\infty^{\mathcal{S}} = \infty$.*

In practice there are only a few scope levels (say, 5). The desired effect, as formalized next, is in admitting low-level roads only near the start or target positions until higher level roads become widely available.

Definition 3.2 ([16]). *Let (G, w) be a road network and $x \in V(G)$. An edge $e = (u, v) \in E(G)$ is* x-admissible *in G for a scope mapping $\mathcal{S}$ if, and only if, there exists a walk $P \subseteq G - e$ from x to u such that*

1. *each edge of P is x-admissible in $G - e$ for $\mathcal{S}$,*
2. *P is optimal subject to (1), and*
3. *for $\ell = \mathcal{S}(e)$, $\sum_{f \in E(P),\, \mathcal{S}(f) > \ell} w(f) \leq \nu^{\mathcal{S}}_{\ell}$.*

Definition 3.3 ([16]). *Let (G, w) be a road network and $\mathcal{S}$ a scope mapping. For a walk $P = (s = u_0, e_1, \ldots e_k, u_k)$ in G; P is* s-admissible *in G for $\mathcal{S}$ if every $e_i \in E(P)$ is s-admissible in G for $\mathcal{S}$.*

Static $\mathcal{S}$-Dijkstra's Algorithm. Aforementioned seemingly complicated definitions can be smoothly integrated into (the bidirectional variants of) Dijkstra's or A* algorithms, simply keeping track of the extreme s-admissibility (or t-adm. in reverse dir.) condition:

- For every accessed vertex v and each scope level $\ell \in Im(\mathcal{S})$, the algorithm keeps, as $\sigma_\ell[v]$, the best achieved value of the sum $\sum_{f \in E(P),\, \mathcal{S}(f) > \ell} w(f)$.
- The s-admissibility of edges e starting in v then depends just on $\sigma_{\mathcal{S}(e)}[v] \leq \nu^{\mathcal{S}}_{\mathcal{S}(e)}$, and only s-admissible edges are relaxed further.

Theorem 3.4 ([16]). *$\mathcal{S}$-Dijkstra's algorithm (uni-directional), for a road network (G, w), a scope mapping $\mathcal{S}$, and a start vertex $s \in V(G)$, computes an optimal s-admissible walk from s to every $v \in V(G)$ in time $\mathcal{O}\big(|E(G)| \cdot |Im(\mathcal{S})| + |V(G)| \cdot \log |V(G)|\big)$.*

The most important computational aspect of scope lies in the fact that only the edges of *unbounded* scope level ∞ matter for global preprocessing (an idea related to better known *reach* [5]). Informally, the query algorithm of [16] works in stages: In the *opening cellular phase*, the road network is locally searched (unidirectional $\mathcal{S}$-Dijkstra) from both start and target vertices until only edges of unbounded scope are admissible. Then a small preprocessed "boundary graph" is searched by another algorithm (e.g. hub-based labeling [15]) in the *boundary phase*. Finally, in the *closing cellular phase*, the scope-unbounded long middle section of the route is "unrolled" in the whole network.

We remark that the boundary graph will remain static even in the dynamic scenario (due to expensive preprocessing), and dynamic changes will be mainly dealt with in the closing cellular phase. Yet we have to pay attention to scope admissibility since it is the key to much improved preprocessing [16] to the boundary graph. It is therefore essential to "dynamize" our definition and $\mathcal{S}$-Dijkstra's algorithm for that purpose.

4 $\mathcal{S}$-Dijkstra's Algorithm – Dynamization

In the rest, due to limited space, we only briefly sketch the uni-directional *Dynamic $\mathcal{S}$-Dijkstra's Algorithm* used locally in the opening cellular phase (while the admissibility definition is implicitly embedded in it). This procedure can be routinely turned into bidirectional and then used to resolve dynamic changes in cells during the closing cellular phase.

We first remark on the "only negative change" assumption of our approach (Sec. 2). This well corresponds with a real-world situation in which just "bad things happen on the road", and the driver thus usually has to find an available detour, instead of looking for unlikely road improvements. Therefore, we are content if our query algorithm finds that an optimal route of the original network (wrt. w) is admissible, though not perfectly optimal,[1] in the changed network (w^*). However, when things go worse with w^*, then our algorithm will always find an optimal dynamic-scope admissible detour in the changed network.

Main Informal Idea. Imagine a driver approaching a road restriction or closure. What would she do? Intuitively, the best solution is for her to slip off the original route (even ahead of the restriction), and re-allow the use of low-level (i.e., inadmissible in the ordinary setting) roads nearby the restriction. Of course, she still wants to minimize detour costs and drive reasonably in terms of such adjusted scope admissibility.

Triple Search. Dynamic changes in our $\mathcal{S}$-Dijkstra's algorithm, starting from s, are resolved by a *detour procedure* executed whenever an s-admissible changed edge $e = (u, v)$, i.e. with $w(e) < w^*(e)$, is going to be relaxed. For simplification we assume that there is only one such changed edge e in the whole network.

The detour procedure is analogical to ordinary Dijkstra, except that to its single (called *live*) search it adds two other auxiliary searches called *dead* and *detour*. Their roles are as follows:

- *Live* (the original) – running as in static $\mathcal{S}$-Dijkstra, relaxes only s-admissible edges while using dynamic w^*. Let Q_{live} denote its queue of discovered vertices, d_{live} its temporary distance estimates, and σ_{live} its scope condition vector.
- *Dead* – started from the end v of e as if $w(e)$ was not changed. So, initially, $Q_{dead} = \{v\}$ and $d_{dead} = d_{live}$ except $d_{dead}[v] = d_{live}[u] + w(e)$. The purpose of Q_{dead} is to later identify which alternative walks are actual detours for e.
- *Detour* - the core new search started from u. Upon reaching e, it resets $\sigma_{detour}[u]$ to 0 on all scope levels $\leq \mathcal{S}(e)$. Then it fills Q_{detour} with u and vertices x on the access route from s to u such that a reverse search from u to x does not exhaust σ_{detour} yet. Note that σ_{detour} is normally updated with this reverse search. However, $d_{detour}[x] = d_{live}[x]$ for those added $x \in Q_{detour}$.

[1] Note that a designated detour of a road construction may perhaps turn out faster than another previously optimal route.

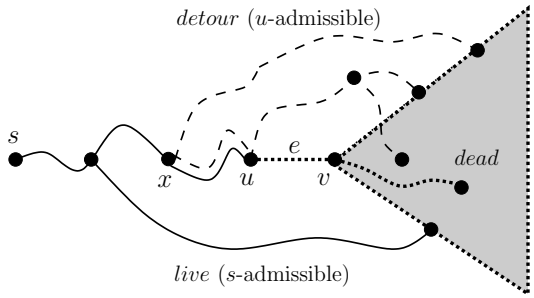

Fig. 1. An illustration of our detour procedure. Upon reaching dynamically changed e such that $w^*(e) > w(e)$; the original live search continues wrt. w^* from already reached vertices except v (solid lines), and two new searches are started – the detour search from u and some of its predecesors (in dashed lines) and the dead search continueing after v wrt. w (dotted area).

The search then continues concurrently with all the three queues (so, starting turns will likely be taken by Q_{detour}). Every relaxation from one of the queues is done as in the static $\mathcal{S}$-Dijkstra's algorithm, i.e., updating also the appropriate $\sigma_\bullet$ vector. Rules which relate together the three searches are outlined below.

Live or Dead. Our driver's desire is to get back to her original route represented by the dead search. This happens when the dead search meets either with the live search (no need to bypass the problematic edge e) or with the detour search (a detour is found).

For more details (see also Fig. 1), imagine a vertex $y \in V(G)$ being relaxed from one of the three queues.

- Suppose y is relaxed from Q_{live}. If $d_{dead}[y] \geq d_{live}[y]$ or $d_{detour}[y] \geq d_{live}[y]$, then y is removed from Q_{dead} or Q_{detour}, respectively.
- Suppose y is relaxed from Q_{detour}. If $d_{dead}[y] \leq d_{detour}[y] < d_{live}[y]$, then y is moved from Q_{detour} (implic. with all its descendants) into Q_{live} setting new distance estimate $d_{live}[y] := d_{detour}[y]$.
- Suppose y is relaxed from Q_{dead}. If $d_{detour}[y] \leq d_{dead}[y] < d_{live}[y]$, then again, y is moved into Q_{live} with new distance estimate $d_{detour}[y]$.

Notice that whenever Q_{dead} or Q_{detour} becomes empty, the other one may also be removed and the algorithm then continues as original $\mathcal{S}$-Dijkstra.

Altogether, the above described Dynamic $\mathcal{S}$-Dijkstra's Algorithm adds at most a constant multiplicative factor to the complexity of static $\mathcal{S}$-Dijkstra, and we propose that usually this increase is only by an additive factor (the dead and detour searches restricted to a neighbourhood of e).

Borrowing Scope in Detour. There is one more specific aspect of the aforementioned detour search. We not only want to reset $\sigma_{detour}[u]$ upon reaching changed

e in the forward direction, but we intend to do the same for $\sigma_{detour}[v]$ "backwards". Informally, we would like to allow low-level roads not only to slip off the original route, but also to return to it from a detour. However, this cannot be done simply in a backward search, and so we instead "borrow" a needed scope value for σ_{detour}.

Precisely, the detour search is allowed to relax even non-s-admissible edges, keeping track of the limited scope value *debt* (on each level). This debt must then be repaid "from v" when the detour search meets the dead search (if it is not repaid in full, then this search branch subsequently dies). Again, due to lack of space, we omit further details.

Multiple Changes. The previous dynamic algorithm may be extended to handle multiple changed edges in w^*, too, as we very briefly outline now. We introduce multiple dead and detour searches, each labeled by a set of all changed edges that affected it. In this view, the original live search is actually the dead search with the empty label.

All these concurrent searches are related together by a complex set of rules depending on their label sets (such as, finishing an L-labeled detour of an edge e_1 moves it to the search labeled by $L \setminus \{e_1\}$; etc). We summarize:

Theorem 4.1. Dynamic $\mathcal{S}$-Dijkstra's algorithm *(uni-directional), for a road network* (G, w) *dynamically changed to* (G, w^*)*, a scope mapping* $\mathcal{S}$*, and a start vertex* $s \in V(G)$*, computes a dynamically* s*-admissible walk from* s *to every* $v \in V(G)$*. This computed walk is optimal in* (G, w^*)*, or in* (G, w)*.*[2]

If c *denotes the number of edges* e *such that* $w(e) < w^*(e)$*, then the algorithm runs in time at most* $\mathcal{O}\big(2^c \cdot (|E(G)| \cdot |Im(\mathcal{S})| + |V(G)| \cdot \log |V(G)|)\big)$.

Even though the factor 2^c may look horrible, we believe the actual effect on time complexity is marginal in real-world scenarios with not-so-many dynamic changes (due to typical "locality" of detours). A thorough experimental evaluation of the complexity of our algorithm is the subject of ongoing research.

5 Discussion

We have outlined the current state of our work on dynamization of the scope-base route planning technique [16] for both unexpected and predictable (to some extend) road network changes. Our approach is aimed at a proper relaxation of scope admissibility when a driver approaches changed road segment, by locally re-allowing nearby roads of lower scope level. At the same time we claim that the computed detour minimizes costs and still remains reasonable in terms of scope admissibility. However, formalized algorithm, its complexity analysis, rigorous proof of correctness and most details are omitted due to lack of space.

[2] This slick formulation is to handle the (unlikely in practice) situation when the changed network actually contains a shorter route from s to v which may not be found.

In a summary, we have shown that a scope-based route planning approach with cellular preprocessing [16] can be used not only in static but also in dynamic road networks. Our immediate future work in this direction will include the following points;

- precise definition of dynamic scope admissibility,
- adding new edges and positive dynamic changes,
- incorporating so called maneuvers, and
- experimentally evaluating this dynamic algorithm on real-world map data.

References

1. EEdsger Dijkstra. A note on two problems in connection with graphs. *Numerische Mathematik*, 1:269–271, 1959.
2. Kenneth L Cooke and Eric Halsey. The shortest route through a network with time-dependent internodal transit times. *Journal of Mathematical Analysis and Applications*, 14(3):493 – 498, 1966.
3. Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. Correction to "a formal basis for the heuristic determination of minimum cost paths". *SIGART Bull.*, 1(37):28–29, 1972.
4. Valerie King. Fully dynamic algorithms for maintaining all-pairs shortest paths and transitive closure in digraphs. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, FOCS '99, pages 81–, Washington, DC, USA, 1999. IEEE Computer Society.
5. Ron Gutman. Reach-based routing: A new approach to shortest path algorithms optimized for road networks. In *Proceedings 6th Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 100–111, 2004.
6. Dorothea Wagner, Thomas Willhalm, and Christos D. Zaroliagis. Dynamic shortest paths containers. *Electr. Notes Theor. Comput. Sci.*, 92:65–84, 2004.
7. Ingrid C. M. Flinsenberg. *Route planning algorithms for car navigation.* PhD thesis, Technische Universiteit Eindhoven, 2004.
8. Dorothea Wagner, Thomas Willhalm, and Christos Zaroliagis. Geometric containers for efficient shortest-path computation. *J. Exp. Algorithmics*, 10, December 2005.
9. Andrew V. Goldberg and Chris Harrelson. Computing the shortest path: A* search meets graph theory. In *In Proc. 16th ACM-SIAM Symposium on Discrete Algorithms*, pages 156–165, 2005.
10. Peter Sanders and Dominik Schultes. Engineering highway hierarchies. In *ESA'06: Proceedings of the 14th conference on Annual European Symposium*, pages 804–816, London, UK, 2006. Springer-Verlag.
11. Daniel Delling and Dorothea Wagner. Landmark-based routing in dynamic graphs. In *Proceedings of the 6th international conference on Experimental algorithms*, WEA'07, pages 52–65, Berlin, Heidelberg, 2007. Springer-Verlag.
12. Dominik Schultes and Peter Sanders. Dynamic highway-node routing. In *WEA'07: Proceedings of the 6th international conference on Experimental algorithms*, pages 66–79, Berlin, Heidelberg, 2007. Springer-Verlag.
13. Dominik Schultes. *Route Planning in Road Networks.* PhD thesis, Karlsruhe University, Karlsruhe, Germany, 2008.

14. Daniel Delling. *Engineering and Augmenting Route Planning Algorithms*. PhD thesis, Karlsruhe University, Karlsruhe, Germany, 2009.
15. Ittai Abraham, Daniel Delling, Andrew V. Goldberg, and Renato F. Werneck. A hub-based labeling algorithm for shortest paths in road networks. In *Proceedings of the 10th international conference on Experimental algorithms*, SEA'11, pages 230–241, Berlin, Heidelberg, 2011. Springer-Verlag.
16. Petr Hliněný and Ondrej Moriš. Scope-Based Route Planning. In *ESA'11: Proceedings of the 19th conference on Annual European Symposium*, pages 445–456, Berlin Heidelberg, 2011. Springer-Verlag. arXiv:1101.3182 (preprint).
17. Petr Hliněný and Ondrej Moriš. Generalized Maneuvers in Route Planning. In *MEMICS'11: 7th International Doctoral Workshop*, Berlin Heidelberg, 2012. Springer-Verlag. (to appear), arXiv:1107.0798 (preprint).